III International Conference on Particle-based Methods – Fundamentals and Applications
PARTICLES 2013
M. Bischoff, E. Oñate, D.R.J. Owen, E. Ramm & P. Wriggers (Eds)

OPEN SOURCE DEM–FEM COUPLING

JAN STRÁNSKÝ[1]

[1] Czech Technical University in Prague, Faculty of Civil Engineering
Thákurova 7, 166 29 Prague 6, Czech republic
jan.stransky@fsv.cvut.cz

Key words: DEM, FEM, Coupling, Multi–physic, Open source

Abstract. The finite element method (FEM) and the discrete element method (DEM) are leading strategies for numerical solution of engineering problems of solid phase. Both are applicable in different situations and sometimes can be beneficially coupled. Coupling of two free open source programs, finite element code OOFEM and discrete element code YADE within another open source package MuPIF – multi-physics integration framework, is presented in this contribution. Design of new MuPIF components for some of the basic coupling strategies (surface coupling, volume coupling, multi-scale approach and contact analysis) is described together with application examples.

1 INTRODUCTION

Numerical simulations are an indispensable part of the current engineering and science development. For different engineering areas there are different numerical methods used. In solid phase mechanics, the leading methods are the finite element method (FEM) and the discrete element method (DEM). FEM is rigorously derived from the continuum theory and is being used for the description of deformable continuous bodies, while DEM describes particulate materials, usually modeled by perfectly rigid particles and their interactions determined from fictitious overlaps of these rigid particles.

Usually the solution is performed by a computer program, which is focused on narrower or wider class of problems (such as solid mechanics, fluid dynamics, heat analysis, DEM etc.). If a combination of two classes of problems is required (coupling of mechanical and heat analysis for instance), often it is possible to find a code allowing such approach. However, in some cases, there exists no such program allowing desired problems combination. For instance, it is possible to couple mechanical and heat analysis within the chosen code, but we would like to use a special material model for mechanical analysis, which is not implemented.

One possible approach to solve such situation would be to write a new or extend existing program implementing requested features. Another possible approach would be to use existing independently developed codes, each one focused on specific class of problems,

and "glue" them together. The latter approach is a motivation of MuPIF [1, 2, 3] tool development.

There are countless software programs for both FEM and DEM. Some of them are commercial (usually) without possibility to change the code and adjust the behavior to our requirements (combination with another software for instance). However, there exist programs with open source code, which the user can modify, possibly for coupling with other programs. In the present article, coupling of FEM code OOFEM [4, 5] and DEM code YADE [6, 7] within MuPIF framework is presented.

Both OOFEM and YADE have the core written in C++ (providing efficient execution of time consuming routines), user interface written in Python (modern dynamic object oriented scripting language, providing easy to use scripting while preserving the C++ efficiency) and extensible object oriented architecture allowing independent implementation of new features - new material model or new particle shapes for instance.

MuPIF (multi-physics integration framework) serves to facilitate combination of independently developed programs (via high-level abstract data exchange interface between individual codes).

The summary of existing MuPIF features is introduced in section 2, basic principles of different coupling strategies are explained (from both theoretical and implementation point of view) in section 3 and specific examples for each coupling strategy are presented in section 4.

2 MuPIF

MuPIF, as was already mentioned, serves to facilitate combination of independently developed programs (i.e. to "glue" them together). It provides high-level abstract data exchange interface between individual codes (components). The MuPIF itself is written in Python, modern dynamic interpreted object oriented language. The object oriented structure together with inheritance and polymorphism concepts allow easy adjustment of default implementation to specific situations, while preserving unified interface design and the possibility to reuse newly created code in other contexts or with different program(s). Such context (e.g. aforementioned mechanical and heat coupling) may be coded as a relatively independent application, in MuPIF called "agent". Final application may be composed of several such agents .

The unified interface is the key concept of MuPIF design, therefore each component (computer program to be "glued") needs to implement their own (Python) application interface, which plays a role of connecting link between the component and MuPIF itself. As a consequence, one component may be instantly replaced with any other program providing required interface, so the user can choose specific program according to specific situation - solver speed, material model library etc.

The overall MuPIF design is shown in figure 1. The computational domain is represented by abstract class `Domain` describing geometry of solved problem and also providing services for spatial search etc. Derived class `DomainView` represents a subset of `Domain`,

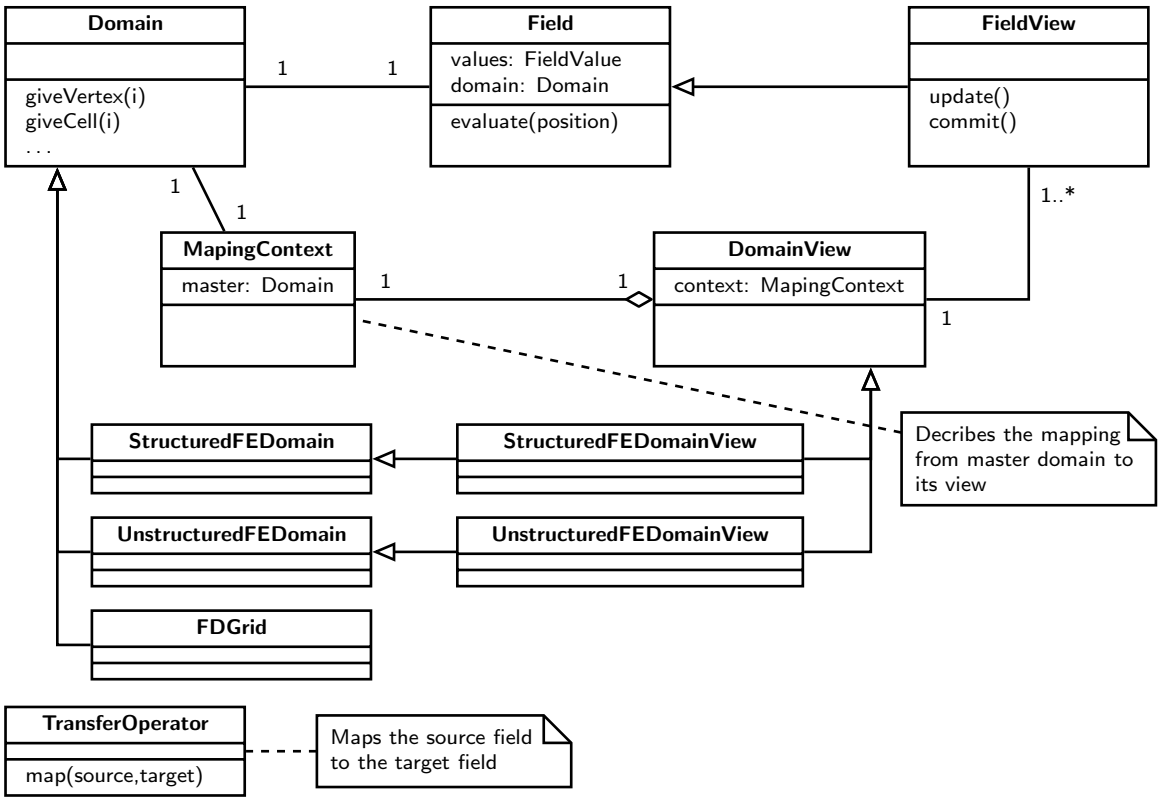


Fig. 1: MuPIF design according to [1]

e.g. boundary, material region etc. `MappingContext` class serves for communication between master `Domain` and corresponding `DomainView`. Any field (solution displacement field or applied body forces field for instance) is represented by `Field` class. Derived `FieldView` class mediates mapping between `Field` defined on master `Domain` and corresponding `DomainView`.

The mutual data exchange between individual components (programs) is performed as an exchange between `Domain`–`Field` pairs. Typically, one application (provider) delivers source `Field`, defined on corresponding `Domain`, while another application (receiver) requires or accepts source `Field`. Each `Domain` may have different discretizations (or even different type of discretization), thus for each problem pair there could be different mapping algorithm. To preserve unified interface, `TransferOperator` with its `map` function is used or overloaded according to specific requirements.

3 DEM–FEM COUPLING

There exist several types of FEM–DEM coupling, e.g. surface coupling [8, 9, 10, 11, 12], volume coupling [13, 14, 15, 16] or multiscale coupling [17]. For each type, different approaches are used, therefore this section is divided into corresponding subsections. In

case of UML diagram, current design is represented with white background, while newly proposed designs are highlighted by gray background.

3.1 MuPIF extension

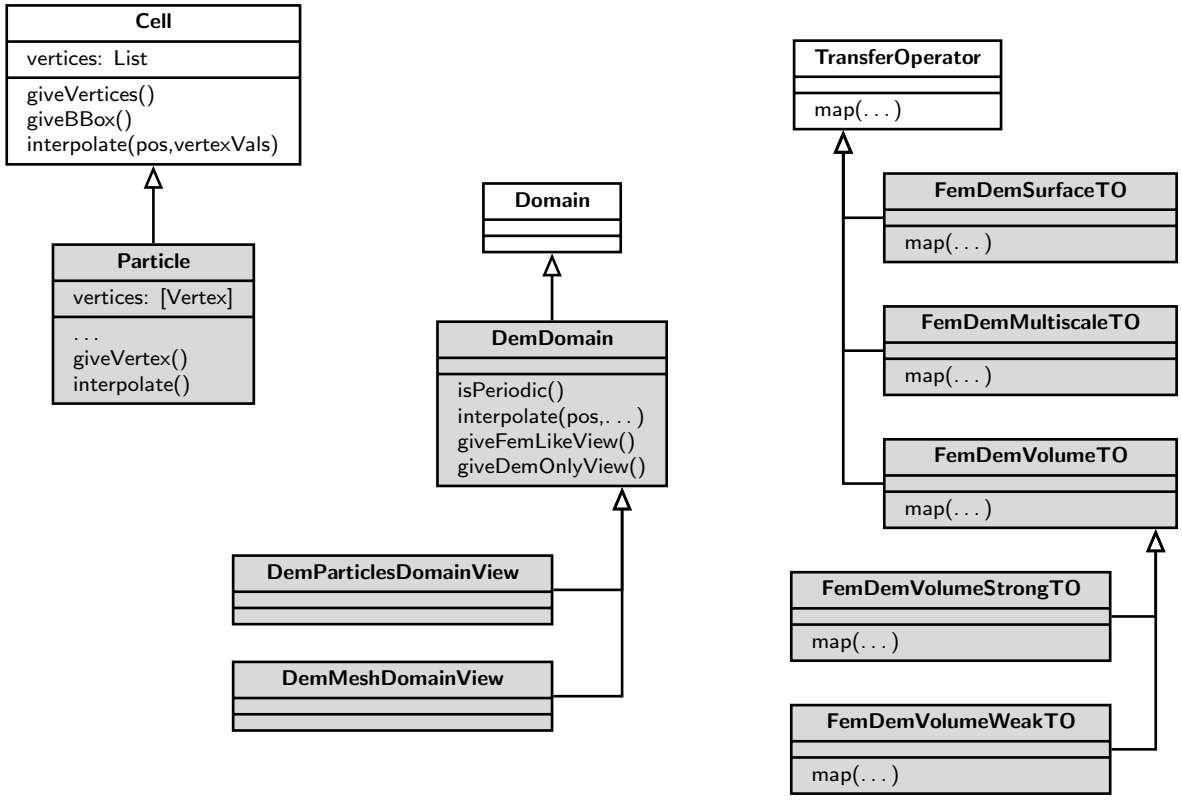


Fig. 2: DEM extension of MuPIF

In this subsection, general features of DEM implementation and differences with the FEM implementation will be discussed. FEM represents the computational domain with finite elements. Usually, the elements have a geometric shape (triangle, tetrahedron, box ...) defined by vertices. Contrary to FEM, DEM represents domain by discrete elements (rigid particles), generally of arbitrary shape. It is also usual in DEM to represent boundaries with triangular particles, which can be rigid or flexible. The triangular particles may be standalone (behaving in the same way as DEM particles) or defined by vertices and connectivity table, behaving more like FEM mesh (at least from geometry point of view). See figure 3 for illustration.

The first difference is `Vertex` and `Cell` classes (from the name, the primary purpose are FEM-like methods). In FEM, `Cell` represents a finite elements defined by several `Vertex` instances. The spatial identifier (bounding box) of the cell is computed as an outer envelope of `Cell`'s vertices. Within the context of DEM particles, `Vertex` represents

the center of the particle, while `Cell` represents the particle itself (a DEM `Cell` has exactly one `Vertex`). Bounding box of such cell is therefore defined by the particle itself. The FEM-like triangular mesh (see above) is more naturally represented by vertices and triangular cells, as it would be in the case of FEM mesh.

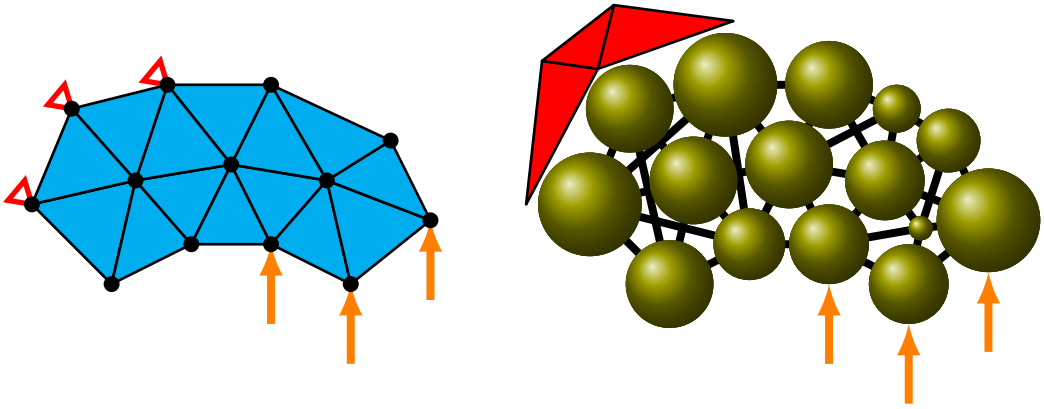


Fig. 3: Schematic illustration of FEM and DEM approach

3.2 Surface coupling

The so called surface coupling [8, 9, 10, 11, 12] is the easiest and most straightforward FEM–DEM coupling strategy. It considers FEM and DEM computational domains as strictly separated.

As an illustrative example, consider a steel beam modeled by FEM falling into an assembly of gravel particles modeled by DEM. Both domains interact with each other, but are physically separated during the entire time of the simulation.

As it is usual in DEM codes to represent boundaries by triangular particles, the natural approach is to use copy of FEM boundary (see `DomainView` in previous section) inside DEM code. As a consequence, the communication between FEM and DEM code would be only within `DomainView` instance(s), on FEM part representing the boundary and on DEM part being composed of triangular particles (representing the same boundary, but from "opposite" part).

In DEM simulation, particles and boundary (represented by triangular particles) may overlap, causing repulsive forces. Such forces are transfered from DEM to FEM code and play the role of load (natural, Neumann) boundary conditions. The displacement is computed and transfered back to DEM code, thus playing the role of displacement (essential, Dirichlet) boundary condition.

For the purpose of surface coupling itself, new class `FemDemSurfaceTO`, derived from `TransferOperator`, was implemented. It "hides" all the implementation, i.e. it provides the functionality described above in its `map` function. Furthermore, a new MuPIF agent specialized on surface FEM–DEM coupling was implemented, providing simple interface

for this task. As was already mentioned, both the FEM and the DEM codes may be replaced by any other program (assuming that the new program already has got proper MuPIF interface).

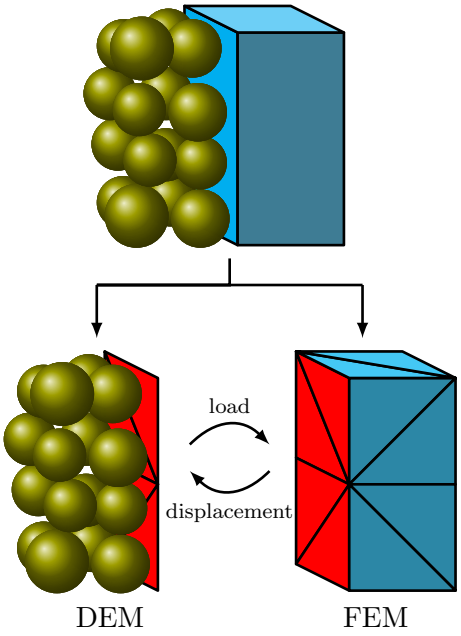


Fig. 4: Surface coupling illustration

3.3 Volume coupling

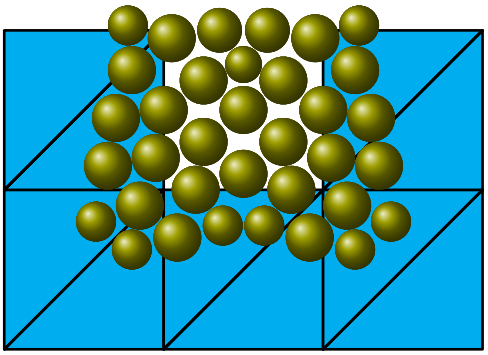


Fig. 5: Volume coupling illustration

Volume coupling (see figure 5 for illustration) is another possible combination of FEM and DEM simulations. It is similar to the surface coupling, but considers certain overlap between both domains. The overlapping parts may have coinciding "vertices", but in general case the overlapping domains may be independent on each other.

Another variety of subtypes of volume coupling is produced by the enforcement of compatibility conditions, resulting in strong or weak coupling [14]. In the case of strong coupling, the overlapping particles are fixed to the elements (using "master/slave" or "hanging nodes" approach), while in the other case (weak coupling or "Arlequin" method), the compatibility and/or equilibrium conditions are satisfied in weak sense [13, 16]. MuPIF is designed such that similar strong/weak coupling may be easily implemented, see [1].

The possible usage of this approach could be a model of concrete beam subjected to an impact load (blast for example). The whole beam would be modeled by FEM and only a small volume of the concrete (the volume to be fragmented and crushed) would be modeled by DEM. To preserve continuous nature of the beam, a transition zone (containing both FEM and DEM) would be included.

From the implementation point of view, the methodology is the same as in the case of surface coupling, i.e. subclassing `TransferOperator` (possibly covering all aforementioned volume coupling types) and creating independent agent for FEM–DEM volume coupling.

3.4 Multiscale coupling

The idea of multiscale simulations is to model the problem on the large (macro) scale using information from a lower (micro) scale. In the current context, the (first order) homogenization [18] is presented. Geometric information (strain) from macro scale (integration points of FEM mesh) is transfered to the micro scale (representative volume element - RVE - modeled by DEM), see figure 6. On the micro scale, the boundary value problem (BVP) governed by the transfered prescribed strain is solved using periodic boundary conditions [19]. The output of the micro-scale problem is the stress tensor (sufficient for explicit solution scheme) and possibly also the constitutive characteristics (stiffness tensor, needed by implicit solution schemes), which are transfered back to the macro-scale problem.

As an example of such approach, consider a sample of sand. In reality, it is composed of individual grains, therefore DEM could be the right modeling approach. However, because of very high computational costs of DEM, the sand is considered as continuum from macroscopic point of view and FEM is used for macroscopic description. To preserve particular nature of the sample, stress–strain law in each integration point is determined not from predefined formulas, but rather from microscale simulations performed on smaller sand samples solved by DEM (thus we do not need any explicit material model for), see figure 6.

From the implementation point of view, design of such approach is already implemented in MuPIF (see [1]), so the DEM code itself only needs to support periodic boundary conditions and stress (possibly also stiffness) estimation [20, 21, 22]. Again, as a final implementation step, new agent for FEM–DEM multiscale coupling was created.

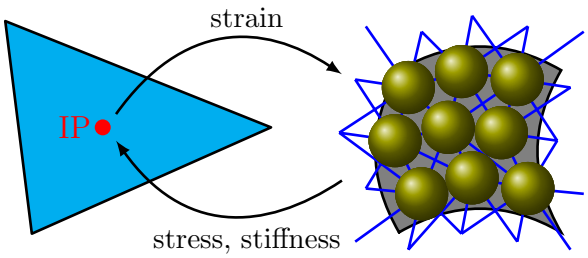


Fig. 6: Multiscale coupling illustration

3.5 Contact coupling

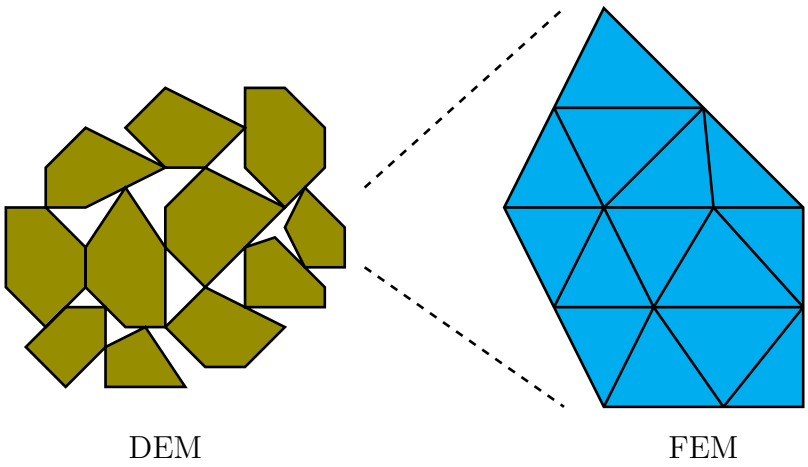
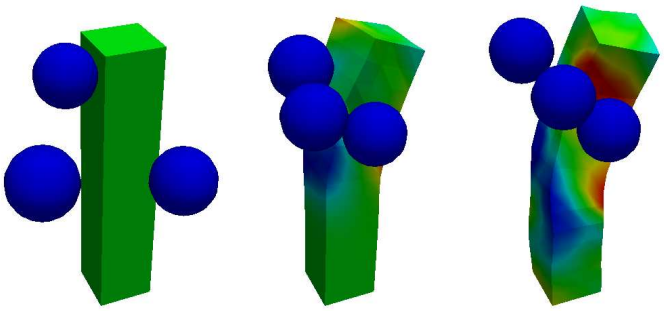


Fig. 7: Illustration of contact approach

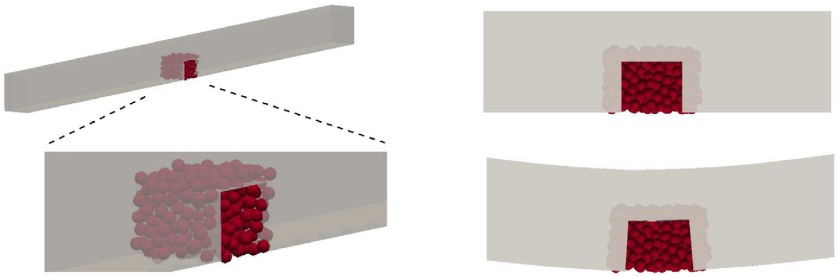
The idea of contact analysis [23] is very simple and opposite to the multiscale approach. The material on the large scale is considered to be of a particulate nature and is modeled by particles using DEM. Each such particle is further modeled by FEM.

There is no strict border between the cases when the solution can be considered as contact FEM analysis and when it is already DEM. For only a few particles we would probably use the former one, but when the number of particles increases, the DEM modeling (with its efficient contact detection algorithms) would be more appropriate. This strategy can be actually considered as full FEM, only the contact detection is "borrowed" from the DEM program. See figure 7 for the illustration.

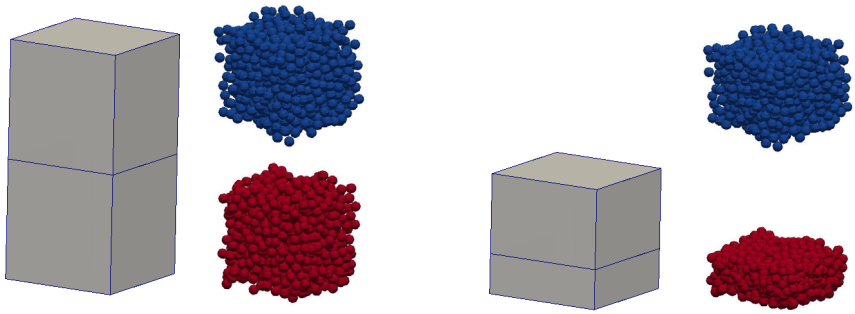
As an example, consider a trabecular bone subjected to compression. The material of trabeculas would be described most likely by non-trivial material model using FEM as a solution tool. However, during compression and compaction, new contacts between



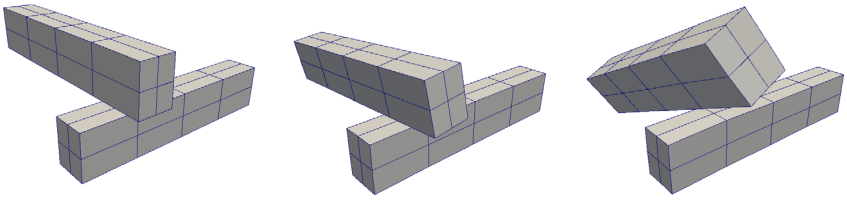
surface coupling



volume coupling



multiscale coupling



contact coupling

Fig. 8: Illustration of described methods

individual trabeculas would occur, which would be solved by DEM solver. Again, the border between FEM contact analysis and this kind of FEM–DEM coupling is not strictly defined.

4 EXAMPLES

In figure 8, some particular pictures of examples (without any detailed description) of described methods are shown. Some examples are (for time reasons) only computed using pure python, without full integration into MuPIF. By the time of the conference, all examples should be included in MuPIF source code and also published on web pages of related projects [3, 5, 7] together with detailed description of simulated task.

5 CONCLUSION AND FUTURE WORK

A new design of integration of discrete-based methods (specifically the discrete element method) into MuPIF framework was proposed. It covers all major DEM–FEM coupling strategies (surface, volume, multiscale and contact), providing for each strategy its "agent" (simple one–purpose application) and related derived classes. Although specific softwares OOFEM and YADE was used, the design is focused on independence on particular program and only features of the discrete element method itself were considered.

Despite the effort of the author, the implementation, documentation, code publication and examples have not been fully completed before this paper deadline. However, they will be presented during the conference and on web pages of related projects (MuPIF, OOFEM and YADE) as soon as they are finished.

ACKNOWLEDGEMENT

Financial support of the Czech Technical University in Prague under project SGS13/034/OHK1/1T/11 is gratefully acknowledged.

REFERENCES

[1] Patzák, B., Rypl, D. and Kruis, J. MuPIF – A distributed multi-physics integration tool. *Adv. Eng. Softw.* (2012) **60–61**:89–97.

[2] Patzák, B. MuPIF: A Distributed Multi-Physics Integration Tool. In: *Proceedings of the Second International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering* (P. Ivnyi and B. H. V. Topping eds.). Stirlingshire, United Kingdom, paper 15. (2011).

[3] MuPIF project web pages. http://mech.fsv.cvut.cz/mupif

[4] Patzák, B. and Bittnar, Z. Design of object oriented finite element code. *Adv. Eng. Softw.* (2001) **32**:pp.759–767.

[5] OOFEM project web pages. http://www.oofem.org

[6] Šmilauer, V., Catalano, E., Chareyre, B., Dorofeenko, S., Duriez, J., Gladky, A., Kozicki, J., Modenese, C., Scholtès, L., Sibille, L., Stránský, J. and Thoeni, K. (2010) Yade Documentation (V. Šmilauer ed.), The Yade Project, 1st ed. http://yade-dem.org/doc/.

[7] YADE project web pages. http://yade-dem.org

[8] Munjiza, A. *The combined finite-discrete element method*. Wiley, Chichester, (2004).

[9] Oñnate, E. and Rojek, J. Combination of discrete element and finite element methods for dynamic analysis of geomechanics problems. *Comput. Method. Appl. M.* (2004) **193**:3087–3128.

[10] Villard, P., Chevalier, B., Le Hello, B. and Combe, G. Coupling between finite and discrete element methods for the modelling of earth structures reinforced by geosynthetic. *Computers and Geotechnics.* (2009) **36**:709–717.

[11] Fakhimi, A. A hybrid discrete–finite element model for numerical simulation of geomaterials. *Computers and Geotechnics.* (2009) **36**:386–395.

[12] Nakashima, H. and Oida, A. Algorithm and implementation of soil–tire contact analysis code based on dynamic FE–DE method. *J. Terramechanics.* (2004) **41**:127–137.

[13] Rousseau, J., Frangin, E., Marin, P. and Daudeville, L. Multidomain finite and discrete elements method for impact analysis of a concrete structure. *Eng. Struct.* (2009) **31**:2735–2743.

[14] Xu, M., Gracie, R. and Belytschko, T. Multiscale Modeling with Extended Bridging Domain Method. In: *Bridging the Scales in Science and Engineering* (J. Fish ed.). Oxford University Press. (2002).

[15] Azvedo, N. M. and Lemos, L. V. Hybrid discrete element/finite element method for fracture analysis. *Comput. Meth. Appl. M.* (2006) **195**:pp.4579–4593.

[16] Wellmann, C. and Wriggers, P. A two-scale model of granular materials. *Comput. Method. Appl. M.* (2012) **205–208**:46–58.

[17] Rojek, J. and Oñate, E. Multiscale analysis using a coupled discrete/finite element model. *Interaction and Multiscale Mechanics* (2007) **1**:1–31.

[18] Geers, M. G. D., Kouznetsova, V. and Brekelmans, W. A. M. Multi-scale computational homogenization: Trends and challenges. *J. Comput. Appl. Math.* (2010) **234**:2175–2182.

[19] Stránský, J. and Jirásek, M. Calibration of particle-based models using cells with periodic boundary conditions. In: *Proc. II International Conference on Particle-based Methods - Fundamentals and Applications* (E. Oñate and D.R.J. Owen eds.). CIMNE, Barcelona, pp. 274–285. (2010).

[20] Kuhl, E., DAddetta, G. A., Leukart, M. and Ramm, E. Microplane modelling and particle modelling of cohesive-frictional materials. In: *Continuous and Discontinuous Modelling of Cohesive-Frictional Materials* (P. A. Veemer et al. eds). Springer, Berlin, pp. 31–46 (2001).

[21] Bagi, K. Stress and strain in granular assemblies. *Mech. Mater.* (1996) **22**:165–177.

[22] Chang, C. S. and Kuhn M. R. On virtual work and stress in granular media. *Int. J. Solids Struc.* (2005) **42**:3773–3793.

[23] Frenning, D. An efficient finite/discrete element procedure for simulating compression of 3D particle assemblies. *Comput. Meth. Appl. M.* (2008) **197**:4266–4272.